This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 035 475 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
13.09.2000 Bulletin 2000/37

(51) Int. Cl.[7]: **G06F 12/02**, G06F 12/14

(21) Application number: **00200739.1**

(22) Date of filing: **02.03.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **05.03.1999 US 263704**

(71) Applicant: **Sun Microsystems Inc.**
Palo Alto, California 94303 (US)

(72) Inventors:
- **Shaylor, Nicholas**
  Newark, CA 94560 (US)
- **Chan, Jeffrey M. W.**
  Mountain View, CA 94040 (US)
- **Oblock, Gary R.**
  Mountain View, CA 94040 (US)
- **Tremblay, Marc**
  Menlo Park, CA 94025 (US)

(74) Representative:
Hanna, Peter William Derek et al
Tomkins & Co.,
5 Dartmouth Road
Dublin 6 (IE)

(54) Simple high-performance memory management unit

(57) A method and an apparatus for translating a virtual address (104) to a physical address (115) in a computer system are described. The system receives (602) a virtual address during an execution or a fetch of a program instruction. The system determines (604) if the virtual address is in an upper portion or a lower portion of a virtual address space (202). If the virtual address is in the lower portion of the virtual address space, the system adds (606) the virtual address to a first base address (108) to produce the physical address. The system also compares (610) the virtual address against an upper bound (110). If the virtual address has a larger value than the upper bound, the system indicates (614) an illegal access. If the virtual address is in the upper portion of the virtual address space, the system adds (608) the virtual address to a second base address (112) to produce the physical address. The system also compares (612) the virtual address against a lower bound (114). If the virtual address has a lower value than the lower bound, the system indicates (~~616~~) that the access is illegal. Thus, the system provides protection from illegal memory accesses. According to one aspect of the present invention, the system determines (618,620) if the virtual address falls within portion of the virtual address space that is protected from write accesses. If so, the system disallows write accesses to the virtual address. Thus, the present invention dispenses with paging and reduces the virtual-to-physical address translation process to a simple addition operation. This leads to faster processor clock speeds, and can greatly reduce the cost of designing and fabricating a computer system.

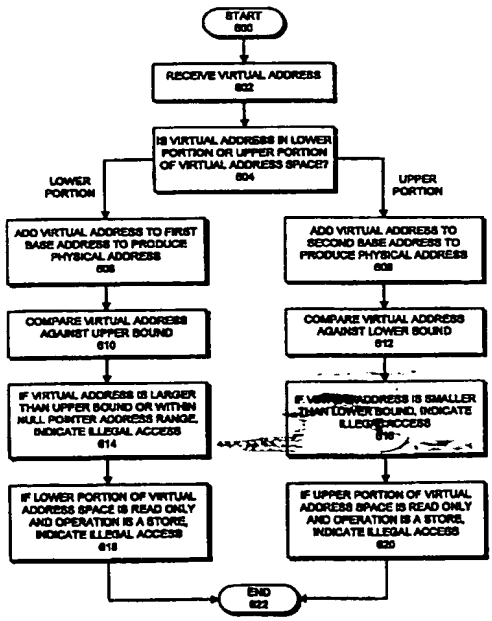

FIG. 6

EP 1 035 475 A1

Description

BACKGROUND

Field of the Invention

[0001] The present invention relates to memory management units for converting virtual addresses to physical addresses in computer systems. More particularly, the present invention relates to a simple, efficient and fast method and apparatus for converting virtual addresses to physical addresses that provides protection from illegal memory accesses.

Related Art

[0002] Modern computer systems typically include hardware and operating system mechanisms to support memory management functions, such as virtual memory and paging. Over the years these systems have grown in complexity to include hardware structures such as memory management units (MMUs) and translation lookaside buffers (TLBs) as well as corresponding operating system mechanisms to support virtual memory and paging.

[0003] This memory management circuitry presently comprises a significant portion of the computational circuitry in a computer system. Consequently, the task of designing this memory management circuitry consumes a great deal of engineering resources, which increases the time and expense involved in developing a computer system. Furthermore, integrating this additional memory management circuitry onto a semiconductor chip increases the die size of the chip. This can lower yield during chip fabrication, which can greatly increase system cost. The additional circuitry also requires circuit signals to traverse larger distances, which can reduce processor clock speeds or result in increased latencies for memory accesses, and thereby reduce system performance. This results in increased latencies for memory accesses. Conventional memory management units also typically include circuitry to implement content addressable memories (CAMs) which consume a great deal of power.

[0004] Memory management functions also comprise a significant portion of the code in an operating system, which can greatly increase the complexity of the operating system. This correspondingly increases the amount of time and expense involved in developing the operating system.

[0005] Furthermore, the performance advantages of paging mechanisms are somewhat diminished for programs written in modern object-oriented programming languages. References to objects tend to be widely dispersed across a large number of pages. Consequently, such object-oriented programs tend to exhibit poor paging performance.

[0006] What is needed is a memory management mechanism that eliminates the complexity of conventional virtual memory paging systems.

SUMMARY

[0007] The present invention provides a method and an apparatus for translating a virtual address to a physical address in a computer system. The system receives a virtual address during an execution or a fetch of a program instruction. The system determines if the virtual address is in an upper portion or a lower portion of a virtual address space. If the virtual address is in the lower portion of the virtual address space, the system adds the virtual address to a first base address to produce the physical address. The system also compares the virtual address against an upper bound. If the virtual address has a larger value than the upper bound, the system indicates an illegal access. If the virtual address is in the upper portion of the virtual address space, the system adds the virtual address to a second base address to produce the physical address. The system also compares the virtual address against a lower bound. If the virtual address has a lower value than the lower bound, the system indicates that the access is illegal. Thus, the system provides protection from illegal memory accesses. According to one aspect of the present invention, the system determines if the virtual address falls within a portion of the virtual address space that is protected from write accesses. If so, the system disallows write accesses to the virtual address. Thus, the present invention dispenses with paging and reduces the virtual-to-physical address translation process to a simple addition operation. This leads to faster processor clock speeds, and can greatly reduce the cost of designing and fabricating a computer system.

BRIEF DESCRIPTION OF THE FIGURES

[0008]

FIG. 1 illustrates a computer system in accordance with an embodiment of the present invention.

FIG. 2 illustrates how regions of a virtual address space are mapped into corresponding regions of a physical address space in accordance with an embodiment of the present invention.

FIG. 3 illustrates some of the circuitry involved in translating a virtual address to a physical address in accordance with an embodiment of the present invention.

FIG. 4 illustrates some of the circuitry involved in detecting an illegal memory access in accordance with an embodiment of the present invention.

FIG. 5 illustrates a computer system with replicated memory management units for servicing instruction references and data references in accordance with an embodiment of the present invention.

FIG. 6 is a flow chart illustrating the process of

translating virtual addresses to physical addresses in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0009] The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

Computer System

[0010] FIG. 1 illustrates a computer system in accordance with an embodiment of the present invention. This computer system includes central processing unit 102, memory management unit (MMU) 106, cache 116, bus 113, memory 120, peripheral device 122, and peripheral device 124. In general, central processing unit 102 may include any type of circuitry for performing computational operations. This includes, but is not limited to, central processing units for mainframe computers, personal computers, workstations, device controllers and smart appliances (such as a smart microwave oven).

[0011] Central processing unit 102 feeds virtual address 104 into MMU 106, which translates virtual address 104 into physical address 115. In the illustrated embodiment, MMU 106 contains six values. The values "read only" 109 (1 bit), upper bound 110 (15 bits) and base address 108 (16 bits) are associated with a lower region of a virtual address space 202. The values "read only" 111 (1 bit), lower bound 114 (15 bits) and base address 112 (16 bits) are associated with an upper region of the virtual address space 202. Note that the values read only 109, upper bound 110 and base address 108 fit within a first 32-bit register, and that the values read only 111, lower bound 114 and base address 112 fit within a second 32-bit register. Packing these values into two 32-bit registers facilitates rapid context switching. The above six values are described in more detail below with reference to FIG. 2.

[0012] Physical address 115 from MMU 106 feeds into cache 116, which stores copies of recently referenced memory items from memory 120. Cache 116 may be any type of cache memory, including separate instruction and data caches. The embodiment illustrated in FIG. 1 includes a unified instruction and data cache that is physically addressed. Note that high-performance computer systems can include a virtually-indexed physically-tagged cache in order to initiate cache accesses in parallel with MMU operations. However, in the illustrated embodiment, MMU 106 has a very simple design involving only a simple add operation, that introduces very little propagation delay. This allows the MMU operation to be performed before the cache lookup without significantly degrading system performance. Another embodiment of the present invention includes a virtually indexed cache memory that operates in parallel with MMU 106.

[0013] Cache 116 is coupled to memory 120 through bus 113. Bus 113 may be any type of communication channel for carrying data between memory 120 and microprocessor chip 100. Bus 113 additionally couples microprocessor chip 100 to peripheral devices 122 and 124. Peripheral devices 122 and 124 may be any type of peripheral device that can be attached a computer system, including, for example, a disk drive or a keyboard. In an alternative embodiment, peripheral devices 122 and 124 are coupled to CPU 102 through a peripheral bus, which is separate from a processor-to-memory bus. Memory 120 may be any type of random access memory for storing code and data for execution and manipulation by central processing unit 102.

Memory Mapping

[0014] FIG. 2 illustrates how regions of a virtual address space 202 are mapped into corresponding regions of a physical address space 220 in accordance with an embodiment of the present invention. In this embodiment, the MMU 106 (see FIG. 1) maps two regions of virtual address space 202 into corresponding regions in physical address space 220.

[0015] Lower region 208 starts near the bottom of the virtual address space 202 and extends upward. Lower region 208 is mapped into a corresponding region in physical address space 220, which extends upward from base address 108 in physical address space 220. The upper limit of lower region 208 is delimited by upper bound 110 in virtual address space 202.

[0016] Upper region 204 starts at the highest addressable location in virtual address space 202 and extends downward. Upper region 204 is mapped into a corresponding region in physical address space 220, which extends downward from base address 112 in physical address space 220. The lower limit of upper region 204 is delimited by lower bound 114 in virtual address space 202. Note that upper region 204 may actually be mapped into a lower area in physical memory than lower region 208.

[0017] In the system illustrated in FIG. 1, each region is associated with a "read only bit," which facilitates providing "read only" protection to the region. In the illustrated embodiment, read only bits 109 and 111 are associated with lower region 208 and upper region 204, respectively. If read only bit 109 is asserted, lower region 208 of virtual address space 202 cannot be mod-

ified. If read only bit 109 is not asserted, lower region 208 virtual address space 202 can be modified. If read only bit 111 is asserted, upper region 204 of virtual address space 202 cannot be modified. If read only bit 111 is not asserted, upper region 204 virtual address space 202 can be modified. These read only bits allow a system to support a simple dynamic linked library mechanism, which allows a library to be "read only" so that it can be shared safely by multiple programs.

[0018] One advantage of the embodiment illustrated in FIGs. 1 and 2 is that the granularity of translation can be coarse. As this coarseness increases, fewer bits of virtual address 104 require modification during the virtual address translation process. This can reduce the size, and hence increase the speed, of an adder that is used during the translation process. Furthermore, as coarseness increases, more lower-order bits of virtual address 104 can be immediately passed into cache 116, which improves cache performance.

[0019] Note that the above arrangement allows relocations to fall on 64K byte boundaries. The two base addresses 108 and 112 contain the upper 16 bits of starting physical addresses of the two regions; the corresponding upper and lower bounds 110 and 114 contain the respective lengths of these two regions.

[0020] In one embodiment of the present invention, an access outside of these two regions causes a hardware trap. During such a hardware trap, MMU 106 is automatically disabled, thereby allowing the trap handling "supervisor" code to reside in unmapped memory. If necessary, the supervisor code can extend the memory bounds.

[0021] Also note that lower region 208 does not extend all the way down to address zero. The first 64K bytes of virtual address space 202 is occupied by null pointer region 210. This provides a mechanism that checks for de-referencing a null pointer. This mechanism operates properly so long as no data structure is more than 64K bytes in length, which is a common restriction. If a data structure is larger that this, a large offset from a null pointer can map onto a valid address within lower region 208.

Address Translating Circuitry

[0022] FIG. 3 illustrates some of the circuitry involved in translating a virtual address 104 into a physical address 115 in accordance with an embodiment of the present invention.

[0023] The circuitry on the left-hand side of FIG. 3 generates virtual address 104. This circuitry includes shifter 306, multiplexer (MUX) 310 and adder 312. Adder 312 adds the contents of register 302 with the contents of either register 304 or immediate field 308. This facilitates array indexing, by storing an array base address register 302 and an array index in either register 304 or immediate field 308.

[0024] Note that immediate field 308 facilitates using a numerical value from an instruction field as an array index. Immediate field 308 may be shifted in shifter 306, which is used to multiply the index by the array element size. This assumes, of course, that the array element size is a power of two. Common array element sizes will be, one byte (no shifting) for byte size array elements, two bytes (shift one bit) for half-word array elements, four bytes (shift two bits) for word-size array elements, or eight bytes (shift three bits) for double precision array elements.

[0025] Once virtual address 104 is constructed, it feeds into the right-hand side of the circuitry in FIG. 3. This circuitry performs the translation from virtual to physical addresses. Note that in this embodiment, the circuitry only operates on the higher order sixteen bits of virtual address 104 and physical address 115, because the lower order sixteen bits virtual address 104 and physical address 115 simply pass through the circuitry without translation.

[0026] The higher order 16 bits of virtual address 104 feed through adder 314, which adds the bits to base address 108 to produce a sum which feeds into an input of multiplexer (MUX) 318.

[0027] The higher order 16 bits of virtual address 104 additionally feed through adder 316, which adds the bits to base address 112 to produce a sum which feeds into the other input of MUX 318.

[0028] The highest order bit of virtual address 104 (bit 31) feeds into the select input of MUX 318. Bit 31 indicates whether virtual address 104 is in upper region 204 or lower region 208 of virtual address space 202. If it is in the upper region 204, the output of adder 316 becomes the output of MUX 318. Otherwise, the output of adder 314 becomes the output of MUX 318.

[0029] The output of MUX 318 becomes the upper 16 bits of physical address 115. Recall that the lower 16 bits of virtual address 104 simply pass through the circuit and become the lower 16 bits of physical address 115.

Circuitry for Detecting Illegal Accesses

[0030] FIG. 4 illustrates some of the circuitry involved in detecting an illegal memory access in accordance with an embodiment of the present invention. This circuitry operates in parallel with the circuitry illustrated in FIG. 3. The circuit illustrated in FIG. 4 includes AND gates 404, 412, 416 and 415, as well as OR gates 408, 414 and 417. The circuit additionally includes compare units 406 and 410.

[0031] Compare units 406 and 410 determine whether virtual address 104 falls below lower bound 114 or exceeds upper bound 110, respectively. Compare unit 406 determines whether virtual address 104 is less than lower bound 114. Compare unit 410 determines whether upper bound 110 is less than virtual address 104.

[0032] AND gates 404 and 412 detect illegal write

operations. AND gate 404 ands together store signal 402 with read only bit 111 from FIG. 1. The output of AND gate 404 is asserted if the operation is a write operation and read only bit 111 is set; this indicates an illegal access. AND gate 412 ands together store signal 402 with read only bit 109. The output of AND gate 412 is asserted if the operation is a write operation and read only bit 109 is set; this similarly indicates an illegal access.

[0033] The outputs of AND gates 404 and 412 and compare units 406 and 410 feed through AND gates 416 and 415, as well as OR gates 408, 414 and 417 to form illegal access signal 418. Illegal access signal 418 is asserted if virtual address 104 is in the upper portion of virtual address space 202 and either AND gate 404 or compare unit 406 is asserted. Illegal access signal 418 is also asserted if virtual address 104 is in the lower portion of virtual address space 202 and either AND gate 412 or compare unit 410 is asserted.

[0034] Note that bit 31 of virtual address 104 and the output of OR gate 408 pass through AND gate 416. This makes the upper half of the circuit in FIG. 4 active only if virtual address 104 is in the upper portion of virtual address space 202. Similarly, the inverse of bit 31 and the output of OR gate 414 pass through AND gate 415. This makes the lower half of the circuit in FIG. 4 active if virtual address 104 is in the lower portion of virtual address space 202.

[0035] Note that the circuitry for detecting null pointer region 210 is not shown. An access to null pointer region 210 can be detected by using another compare unit that compares virtual address 104 against the 64K upper boundary of null pointer region 210. If virtual address 104 falls below this upper boundary, an illegal access is indicated. An access to null pointer region 210 may alternatively be detected by examining the higher order bits past the first 16 bits of virtual address 104. If these bits are all zero values, virtual address 104 falls within null pointer region 210.

[0036] Also note that it is possible to perform bounds checking on physical addresses instead of virtual addresses; this results in precise memory exceptions as opposed to imprecise asynchronous memory exceptions.

Replicated Memory Management Units

[0037] FIG. 5 illustrates a computer system with replicated memory management units for servicing instruction references and data references in accordance with an embodiment of the present invention. In this embodiment, instruction fetch unit 502 within CPU 102 is coupled to instruction cache 510 through MMU 506. This pathway takes care of instruction references. For data references, load store unit 504 within CPU 102 is coupled to data cache 512 through MMU 508.

[0038] Note that MMU 508 is an exact replica of MMU 506, and that these MMUs are kept in consistent states. This means MMU 506 and MMU 508 contain the same values in read only bits 109 and 111, as well as in upper bound 110, lower bound 114, base address 108 and base address 112.

[0039] The simple MMU design of the present invention allows the MMUs to be replicated without taking up large amounts of semiconductor real estate. Also, note that providing multiple MMUs reduces address translation contention between instruction and data references.

Process of Address Translation

[0040] FIG. 6 is a flow chart illustrating the process of translating virtual addresses to physical addresses in accordance with an embodiment of the present invention. The system starts by receiving virtual address 104 during an execution or a fetch of a program instruction (step 602). The system determines if virtual address 104 is an upper portion or a lower portion of virtual address space 202 (step 604). This can be determined by examining the most significant bit of virtual address 104. If the bit is asserted, virtual address 104 is in the upper portion. Otherwise, it is in the lower portion.

[0041] If virtual address 104 is in the lower region 208 of virtual address space 202, the system adds virtual address 104 to base address 108 to produce physical address 115 (step 606). The system also compares virtual address 104 against upper bound 110 (step 610). If virtual address 104 is larger than upper bound 110, the system asserts illegal access signal 418 (step 614). The system also examines read only bit 109. If lower region 208 of virtual address space 202 is "read only" and the access is a store operation, the system also asserts illegal access signal 418 (step 618).

[0042] If virtual address 104 is in the upper region 204 of virtual address space 202, the system adds virtual address 104 to base address 112 to produce physical address 115 (step 608). The system also compares virtual address 104 against lower bound 114 (step 612). If virtual address 104 is smaller than lower bound 114, the system asserts illegal access signal 418 (step 616). The system also examines read only bit 111. If upper region 204 of virtual address space 202 is "read only" and the access is a store operation, the system also asserts illegal access signal 418 (step 620).

[0043] Note that for both the upper region 204 and the lower region 208 the conversion operation is an addition. The conversion for upper region 204 may appear to involve a subtraction operation because upper region 204 starts at the top of virtual address space 202 and offsets proceed downward. However, a downward offset can be implemented with an adder using two's complement arithmetic.

[0044] In the embodiment illustrated in FIG. 5, the test for "read only" is redundant for MMU 506 because MMU 506 handles instructions, which are all "read only."

[0045] The foregoing descriptions of embodiments

of the invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the invention. The scope of the invention is defined by the appended claims.

Claims

1. A method for translating a virtual address (104) into a physical address (115) in a computer system, comprising:

receiving the virtual address during an execution or a fetch of a program instruction;
determining (604) if the virtual address is in an upper portion or a lower portion of a virtual address space (202);
if the virtual address is in the lower portion of the virtual address space,

adding (606) the virtual address to a first base address (108) to produce the physical address, and
comparing (610) the virtual address against an upper bound (110),
if the virtual address has a larger value than the upper bound, indicating (614) that the access is illegal; and

if the virtual address is in the upper portion of the virtual address space,

adding (608) the virtual address to a second base address (112) to produce the physical address, and
comparing (612) the virtual address against a lower bound (114),
if the virtual address has a lower value than the lower bound, indicating (616) that the access is illegal.

2. The method of claim 1, further comprising:

if the virtual address is in the lower portion of the virtual address space, determining if the lower portion of the virtual address space is protected from write accesses, and if so disallowing (618) write accesses to the virtual address; and
if the virtual address is in the upper portion of the virtual address space, determining if the upper portion of the virtual address space is protected from write accesses, and if so disallowing (620) write accesses to the virtual address;

3. The method of claim 1 or claim 2, further comprising disabling address translations during operating system accesses to memory.

4. The method of any one of claims 1 to 3, wherein the upper portion of the virtual address space (202) contains read-only library routines.

5. The method of any one of claims 1 to 4, wherein the program instruction may include an instruction that bypasses a cache (116) in the computer system for direct memory access (DMA) operations.

6. The method of any one of claims 1 to 5, further comprising determining if the access involves a null pointer by comparing the virtual address against a range of null pointer addresses (210), and if the virtual address falls within the range of null pointer addresses, indicating (614) that the access is illegal.

7. The method of any one of claims 1 to 6, wherein the method is performed within a memory management unit (106) that is integrated into a microprocessor chip (100).

8. An apparatus (106) for translating a virtual address (104) into a physical address (115) in a computer system, comprising:

an input that receives the virtual address during an execution or a fetch of a program instruction;
an adder (314,316) that creates a sum of the virtual address and a first base address (108) if the virtual address is in a lower portion of a virtual address space, and that creates a sum of the virtual address and a second base address (112) if the virtual address is in an upper portion of the virtual address space; and
an access validating circuit that indicates an illegal access (418) if the virtual address is in the lower portion of the virtual address space and the virtual address has a larger value than an upper bound (110), and that indicates an illegal access if the virtual address is in the upper portion of the virtual address space and the virtual address has a lower value than a lower bound (114).

9. The apparatus of claim 8, wherein the adder includes:

a first adding circuit (314) for adding the virtual address to the first base address (108);
a second adding circuit (316) for adding the virtual address to the second base address (112), and

a selector circuit (318) that selects between an output of the first adding circuit and an output of the second adding circuit.

10. The apparatus of claim 8 or claim 9, wherein the access validating circuit is configured to determine if the lower portion of the virtual address space is protected from write accesses, and if so to disallow write accesses to the lower portion of the virtual address space, and is configured to determine if the upper portion of the virtual address space is protected from write accesses, and if so to disallow write accesses to the upper portion of the virtual address space.

11. The apparatus of any one of claims 8 to 10, further comprising a mechanism that disables address translations during operating system accesses to memory.

12. The apparatus of any one of claims 8 to 11, wherein the upper portion of the virtual address space (202) contains read-only library routines.

13. The apparatus of any one of claims 8 to 12, further comprising a mechanism that bypasses a cache (116) in the computer system for direct memory access (DMA) operations.

14. The apparatus of any one of claims 8 to 13, wherein the access validating circuit is configured to determine if the access involves a null pointer by comparing the virtual address against a range of null pointer addresses (210), and if the virtual address falls within the range of null pointer addresses, indicating (614) that the access is illegal.

15. The apparatus of any one of claims 8 to 14, wherein the apparatus is integrated into a microprocessor chip (100).

16. A computer system, comprising:

a central processing unit (102);
a main memory (120); and
a first memory management unit (106) for translating a virtual address (104) into a physical address (115), coupled between the central processing unit and the main memory, the first memory management unit comprising,

an adder (314,316) that creates a sum of the virtual address and a first base address if the virtual address is in a lower portion of a virtual address space, and that creates a sum of the virtual address and a second base address if the virtual address is in an upper portion of the virtual address space, and
an access validating circuit that indicates an illegal access (418) if the virtual address is in the lower portion of the virtual address space and the virtual address has a larger value than an upper bound (110), and that indicates an illegal access if the virtual address is in the upper portion of the virtual address space and the virtual address has a lower value than a lower bound (114).

17. The computer system of claim 16, further comprising a cache memory (116) coupled between the central processing unit (102) and the main memory (120), wherein the first memory management unit (106) is coupled between the central processing unit and the cache memory.

19. The computer system of claim 18, further comprising:

an instruction cache (510) coupled between the central processing unit and the main memory;

a data cache (512) coupled between the central processing unit and the main memory; and
a second memory management unit (508) which is configured identically with the first memory management unit (506);
wherein the first memory management unit (506) is coupled between the central processing unit and the instruction cache (510), and the second memory management unit (508) is coupled between the central processing unit and the data cache(512), whereby the first memory management unit performs translations for instruction accesses and the second memory management unit performs translations for data accesses.

20. Computer software or firmware for translating a virtual address (104) into a physical address (115) in a computer system, which when running on a computer is capable of performing the steps of any one of claims 1 to 7.

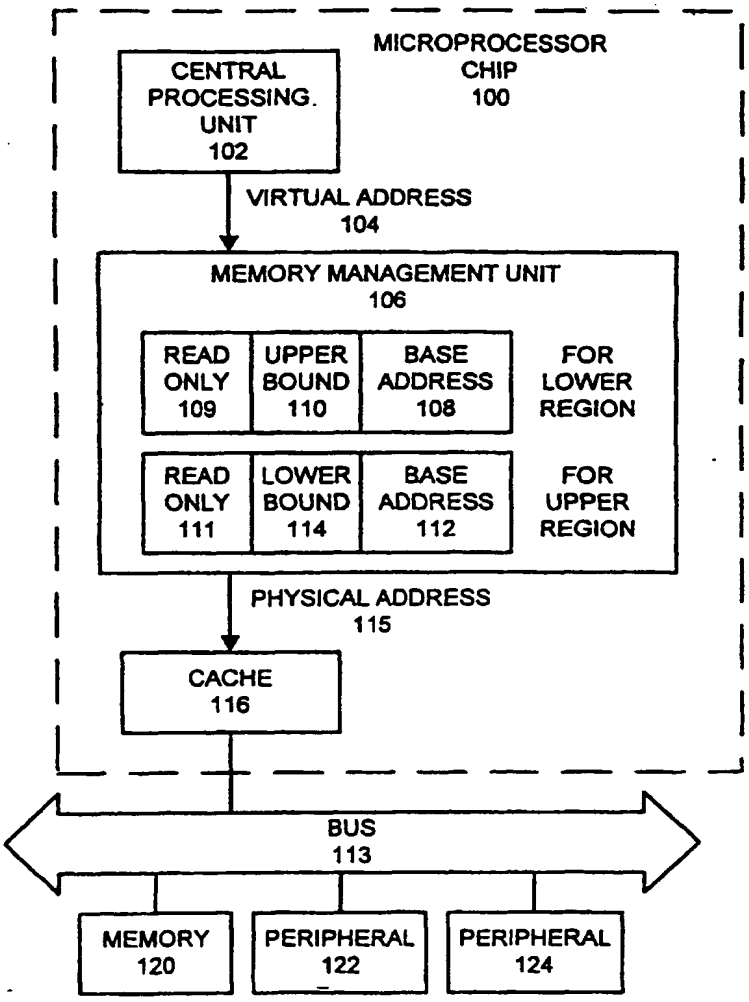
MICROPROCESSOR CHIP 100
CENTRAL PROCESSING. UNIT 102
VIRTUAL ADDRESS 104
MEMORY MANAGEMENT UNIT 106
READ ONLY 109
UPPER BOUND 110
BASE ADDRESS 108
FOR LOWER REGION
READ ONLY 111
LOWER BOUND 114
BASE ADDRESS 112
FOR UPPER REGION
PHYSICAL ADDRESS 115
CACHE 116
BUS 113
MEMORY 120
PERIPHERAL 122
PERIPHERAL 124

FIG. 1

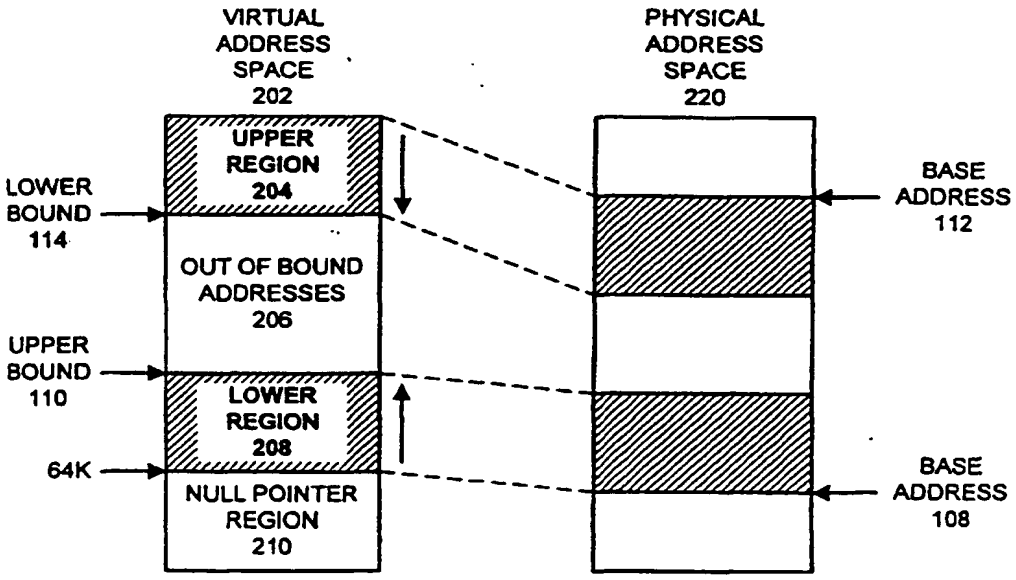
VIRTUAL ADDRESS SPACE 202
PHYSICAL ADDRESS SPACE 220
UPPER REGION 204
LOWER BOUND 114
OUT OF BOUND ADDRESSES 206
UPPER BOUND 110
LOWER REGION 208
64K
NULL POINTER REGION 210
BASE ADDRESS 112
BASE ADDRESS 108

FIG. 2

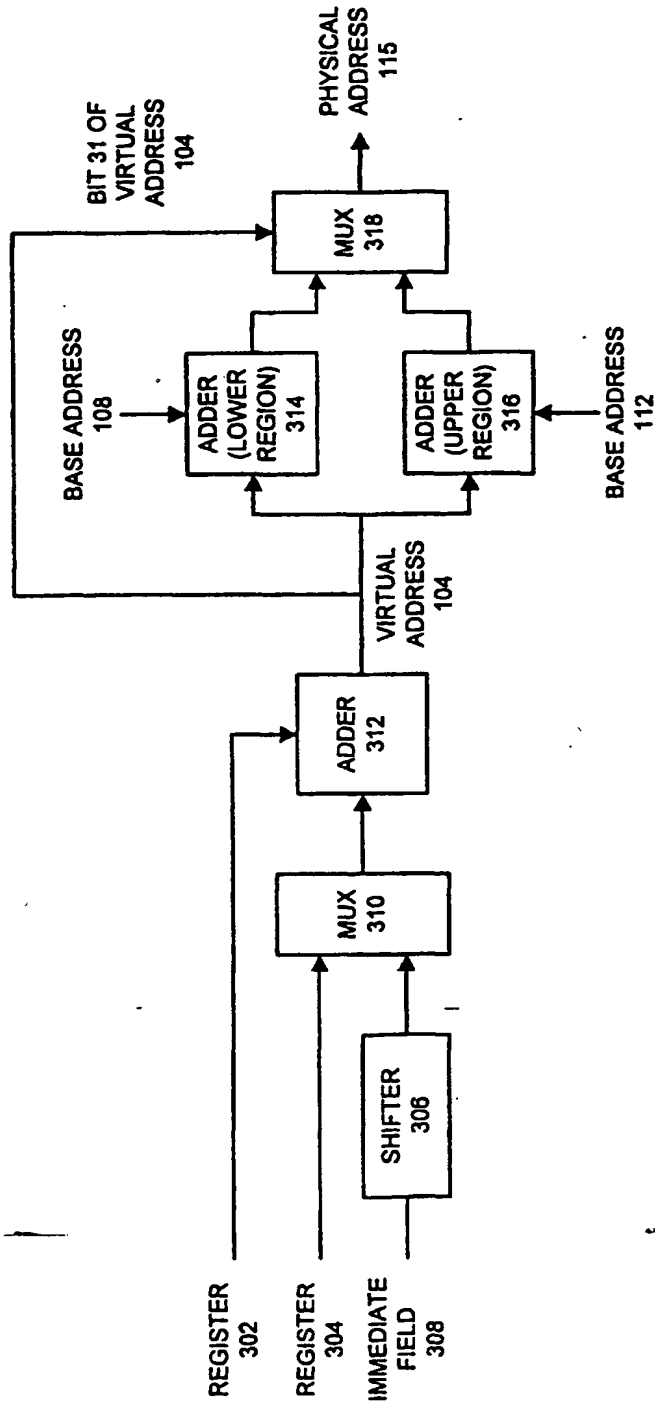
REGISTER 302
REGISTER 304
IMMEDIATE FIELD 308
SHIFTER 306
MUX 310
ADDER 312
VIRTUAL ADDRESS 104
BASE ADDRESS 108
ADDER (LOWER REGION) 314
ADDER (UPPER REGION) 316
BASE ADDRESS 112
BIT 31 OF VIRTUAL ADDRESS 104
MUX 318
PHYSICAL ADDRESS 115

FIG. 3

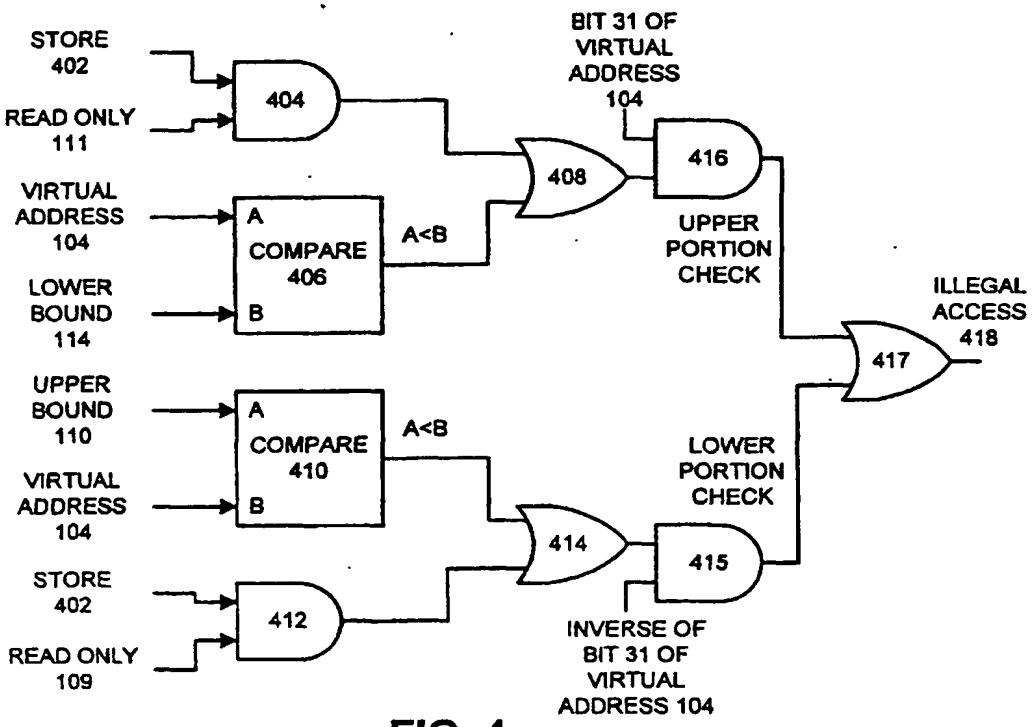
STORE
402
READ ONLY
111
404
BIT 31 OF
VIRTUAL
ADDRESS
104
408
416
VIRTUAL
ADDRESS
104
A
COMPARE
406
B
A<B
UPPER
PORTION
CHECK
LOWER
BOUND
114
ILLEGAL
ACCESS
418
417
UPPER
BOUND
110
A
COMPARE
410
B
A<B
LOWER
PORTION
CHECK
VIRTUAL
ADDRESS
104
414
415
STORE
402
412
READ ONLY
109
INVERSE OF
BIT 31 OF
VIRTUAL
ADDRESS 104

FIG. 4

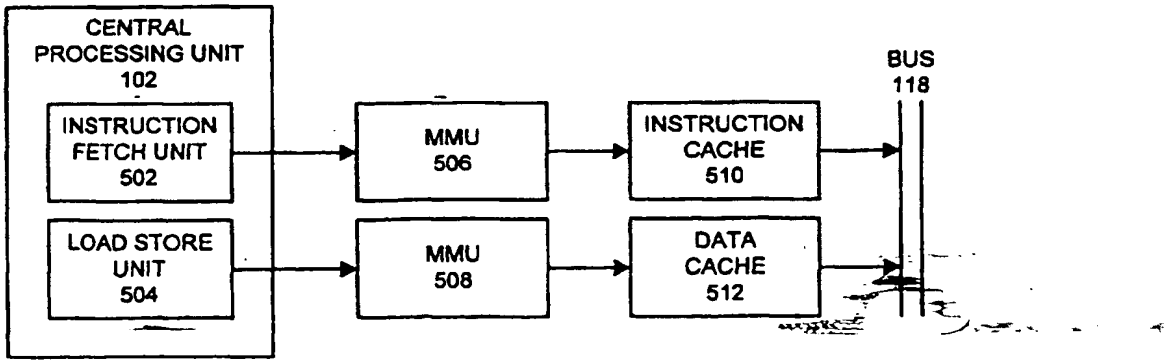
CENTRAL
PROCESSING UNIT
102
INSTRUCTION
FETCH UNIT
502
LOAD STORE
UNIT
504
MMU
506
MMU
508
INSTRUCTION
CACHE
510
DATA
CACHE
512
BUS
118

FIG. 5

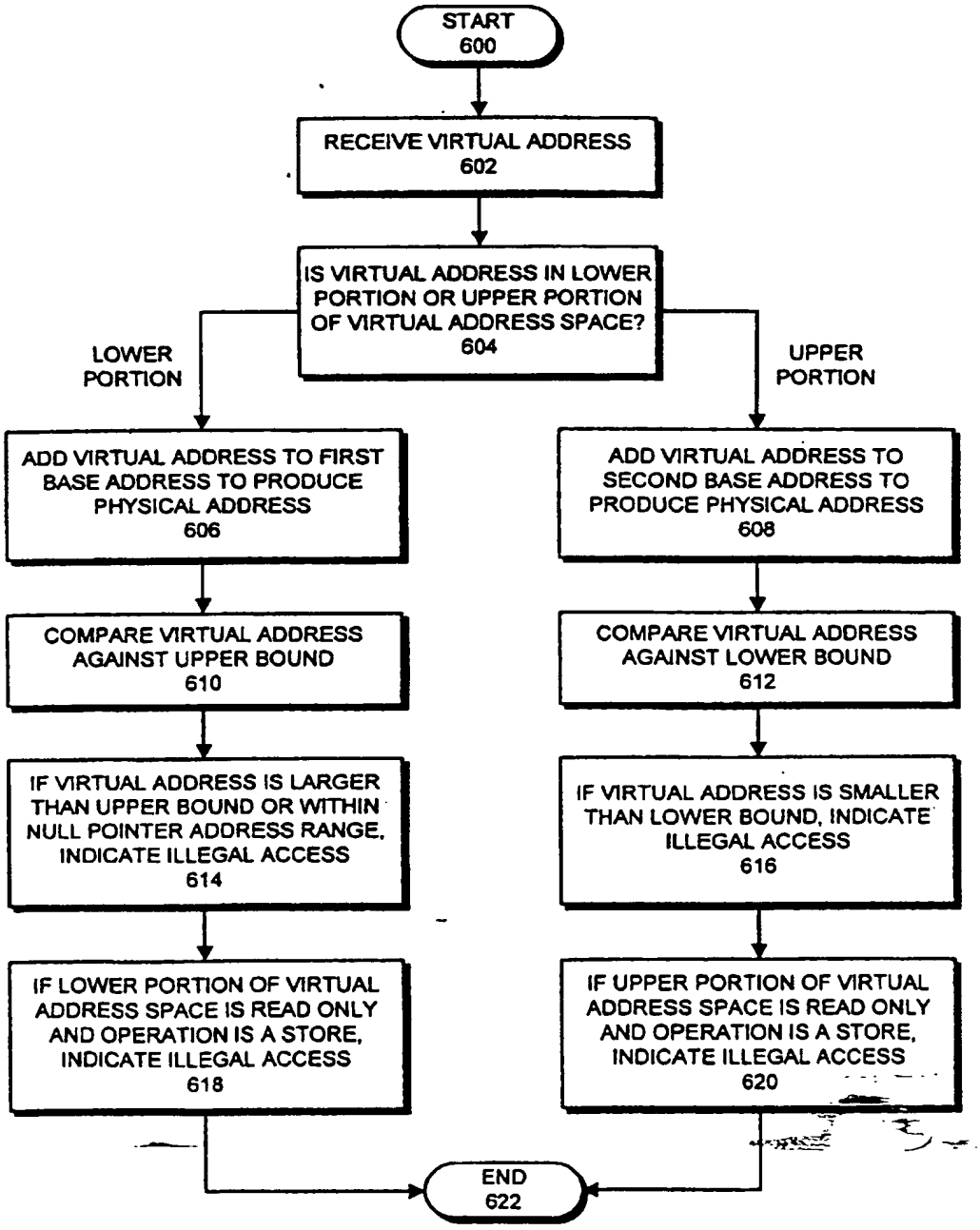
START
600
RECEIVE VIRTUAL ADDRESS
602
IS VIRTUAL ADDRESS IN LOWER PORTION OR UPPER PORTION OF VIRTUAL ADDRESS SPACE?
604
LOWER PORTION
UPPER PORTION
ADD VIRTUAL ADDRESS TO FIRST BASE ADDRESS TO PRODUCE PHYSICAL ADDRESS
606
ADD VIRTUAL ADDRESS TO SECOND BASE ADDRESS TO PRODUCE PHYSICAL ADDRESS
608
COMPARE VIRTUAL ADDRESS AGAINST UPPER BOUND
610
COMPARE VIRTUAL ADDRESS AGAINST LOWER BOUND
612
IF VIRTUAL ADDRESS IS LARGER THAN UPPER BOUND OR WITHIN NULL POINTER ADDRESS RANGE, INDICATE ILLEGAL ACCESS
614
IF VIRTUAL ADDRESS IS SMALLER THAN LOWER BOUND, INDICATE ILLEGAL ACCESS
616
IF LOWER PORTION OF VIRTUAL ADDRESS SPACE IS READ ONLY AND OPERATION IS A STORE, INDICATE ILLEGAL ACCESS
618
IF UPPER PORTION OF VIRTUAL ADDRESS SPACE IS READ ONLY AND OPERATION IS A STORE, INDICATE ILLEGAL ACCESS
620
END
622

FIG. 6

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 20 0739

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	FR 2 766 597 A (INSIDE TECHNOLOGIES) 29 January 1999 (1999-01-29) * page 3, line 16 - page 5, line 14; figure 2 *	1,3,7,8, 11,15,16	G06F12/02 G06F12/14
A	EP 0 656 592 A (MOTOROLA INC) 7 June 1995 (1995-06-07) * column 1, line 46 - column 3, line 1; figures 1,2,4 *	1,2,4,8, 10,11, 15-17,19	
A	US 5 687 342 A (KASS WILLIAM J) 11 November 1997 (1997-11-11) * column 2, line 20 - column 3, line 4; figure 2 *	1,8,9,16	
A	US 3 803 559 A (BANDOO T ET AL) 9 April 1974 (1974-04-09)	1,3,6,8, 11,16	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	17 May 2000	Nielsen, O

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 00 20 0739

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

17-05-2000

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
FR 2766597 A	29-01-1999	NONE	
EP 0656592 A	07-06-1995	US 5623636 A	22-04-1997
		JP 7191903 A	28-07-1995
US 5687342 A	11-11-1997	NONE	
US 3803559 A	09-04-1974	JP 48029327 A	18-04-1973
		JP 51040772 B	05-11-1976

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82